MIS780 Advanced AI For Business - Assignment 2 - T2 2024

Task Number: Business Problem Name

Student Name: *enter your full name here*

Student ID: *enter your student ID here*

Table of Content

1. Executive Summary
2. Data Preprocessing
3. Predictive Modeling
4. Experiments Report

1. Executive Summary

Use this section to introduce the business problem, data set, method, experiments, and obtained results

2. Data Preprocessing

Carry out necessary data preprocessing and exploration.

```
# Importing the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_error

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM
from tensorflow.keras.optimizers import SGD
from tensorflow.random import set_seed

set_seed(455)
np.random.seed(455)
```

```
dataset = pd.read_csv('/Part3_GoldPrice.csv', index_col='Date',
parse_dates=["Date"], dayfirst=True)

print(dataset.head())
print(dataset.describe())
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call
last)
<ipython-input-3-417cbd9a919f> in <cell line: 1>()
----> 1 dataset = pd.read_csv('/Part3_GoldPrice.csv',
index_col='Date', parse_dates=["Date"], dayfirst=True)
      2
      3 print(dataset.head())
      4 print(dataset.describe())

/usr/local/lib/python3.10/dist-packages/pandas/io/parsers/readers.py
in read_csv(filepath_or_buffer, sep, delimiter, header, names,
index_col, usecols, dtype, engine, converters, true_values,
false_values, skipinitialspace, skiprows, skipfooter, nrows,
na_values, keep_default_na, na_filter, verbose, skip_blank_lines,
parse_dates, infer_datetime_format, keep_date_col, date_parser,
date_format, dayfirst, cache_dates, iterator, chunksize, compression,
thousands, decimal, lineterminator, quotechar, quoting, doublequote,
escapechar, comment, encoding, encoding_errors, dialect, on_bad_lines,
delim_whitespace, low_memory, memory_map, float_precision,
storage_options, dtype_backend)
    946     kwds.update(kwds_defaults)
    947
--> 948     return _read(filepath_or_buffer, kwds)
    949
    950

/usr/local/lib/python3.10/dist-packages/pandas/io/parsers/readers.py
in _read(filepath_or_buffer, kwds)
    609
    610     # Create the parser.
--> 611     parser = TextFileReader(filepath_or_buffer, **kwds)
    612
    613     if chunksize or iterator:

/usr/local/lib/python3.10/dist-packages/pandas/io/parsers/readers.py
in __init__(self, f, engine, **kwds)
   1446
   1447         self.handles: IOHandles | None = None
-> 1448         self._engine = self._make_engine(f, self.engine)
   1449
   1450     def close(self) -> None:
```

```
/usr/local/lib/python3.10/dist-packages/pandas/io/parsers/readers.py
in _make_engine(self, f, engine)
   1703                 if "b" not in mode:
   1704                     mode += "b"
-> 1705             self.handles = get_handle(
   1706                 f,
   1707                 mode,

/usr/local/lib/python3.10/dist-packages/pandas/io/common.py in
get_handle(path_or_buf, mode, encoding, compression, memory_map,
is_text, errors, storage_options)
    861         if ioargs.encoding and "b" not in ioargs.mode:
    862             # Encoding
--> 863             handle = open(
    864                 handle,
    865                 ioargs.mode,

FileNotFoundError: [Errno 2] No such file or directory:
'/Part3_GoldPrice.csv'
```

```
dataset.info()
print(dataset.dtypes)

dataset.isna().sum()

# Plot the dataset to visualize gold prices in various currencies
fig, axs = plt.subplots(4, 1, figsize=(10, 12))
axs[0].plot(dataset['USD'], 'tab:green')
axs[0].set_title('Gold Price in USD')
axs[1].plot(dataset['EUR'], 'tab:blue')
axs[1].set_title('Gold Price in EUR')
axs[2].plot(dataset['GBP'], 'tab:grey')
axs[2].set_title('Gold Price in GBP')
axs[3].plot(dataset['INR'], 'tab:red')
axs[3].set_title('Gold Price in INR')

# Splitting the dataset
tstart = 1985
tend = 2021

def train_test_split(dataset, tstart, tend):
    train = dataset.loc[f"{tstart}":f"{tend}", "USD"]
    test = dataset.loc[f"{tend+1}":, "USD"]
    return train, test

training_set, test_set = train_test_split(dataset, tstart, tend)

print('Training set shape:', training_set.shape)
print('Test set shape:', test_set.shape)
```

```
plt.figure(figsize=(16, 5))
plt.plot(training_set, label='Train')
plt.plot(test_set, label='Test')
plt.title('Gold Price in USD Over Time')
plt.ylabel('Gold Price (USD)')
plt.xlabel('Time')
plt.legend(loc='upper left')
plt.show()

# Scaling the data
sc = MinMaxScaler(feature_range=(0, 1))
training_set = training_set.values.reshape(-1, 1)
training_set_scaled = sc.fit_transform(training_set)
print('Training set scaled shape:', training_set_scaled.shape)
training_set_scaled =
training_set_scaled.reshape(training_set_shape[0],
training_set_shape[1])
print('training_set_scaled shape:', training_set_scaled.shape)

n_steps = 50  # Number of time steps (windows)
forecasting_horizon = 14  # 2-week forecasting horizon

# Split the training set into samples
X_train, y_train = split_sequence(training_set_scaled, n_steps,
forecasting_horizon)
y_train = y_train.reshape(y_train.shape[0], y_train.shape[1], 1)

print('X_train shape:', X_train.shape)
print('y_train shape:', y_train.shape)

plt.figure(figsize=(7, 4))
plt.plot(np.arange(1, n_steps+1, 1), X_train[0,:,0], label='USD')

plt.plot(np.arange(n_steps, n_steps+forecasting_horizon, 1),
y_train[0], label='Predicted Price (y_train)')

plt.title('One Training Sample')
plt.ylabel('Gold Price (USD)')
plt.xlabel('Time')
plt.legend(loc='lower right')
plt.show()
```

3. Predictive Modeling

Create and explain your models (e.g., model architecture, model parameters). Evaluate the models on the experimental data sets.

```
# Build the LSTM model
model_lstm = Sequential()
model_lstm.add(LSTM(units=100, activation="tanh",
input_shape=(n_steps, 1)))
model_lstm.add(Dense(units=forecasting_horizon))  # Forecast for 2
weeks

# Compile the model
model_lstm.compile(optimizer="RMSprop", loss="mse")

model_lstm.summary()

# Train the model
model_lstm.fit(X_train, y_train, epochs=100, batch_size=32)

# Rescale the test set
inputs = test_set.values.reshape(-1, 1)
inputs_scaled = sc.transform(inputs)

# Reshape the test set back to original format after rescaling
X_test, y_test = split_sequence(inputs_scaled, n_steps,
forecasting_horizon)

#prediction
predicted_gold_price = model_lstm.predict(X_test)

# Inverse transform the predicted values and the test labels
predicted_gold_price = sc.inverse_transform(predicted_gold_price)
y_test = sc.inverse_transform(y_test.reshape(-1,
1)).reshape(y_test.shape)

print('Predicted gold price shape: ', predicted_gold_price.shape)

def return_mae(test, predicted):
    mae = mean_absolute_error(test, predicted)
    print("Mean Absolute Error {:.2f}.".format(mae))

for i in range(forecasting_horizon):
    print(f"Forecasting Horizon {i+1} day(s):")
    return_mae(y_test[:, i], predicted_gold_price[:, i])

#
X_test = X_test.reshape(-1, 50)
X_test = sc.inverse_transform(X_test)
X_test = X_test.reshape(number_test_samples, 50,4)

#
sample_index = 6

plt.figure(figsize=(7, 4))
plt.plot(np.arange(1, n_steps+1, 1),X_test[sample_index,:,1])
```

```
plt.plot(np.arange(n_steps, n_steps+forecasting_horizon,
1),y_test[sample_index,:])
plt.plot(np.arange(n_steps, n_steps+forecasting_horizon,
1),predicted_stock_price[sample_index,:])
plt.title('Testing Sample ' + str(sample_index))
plt.ylabel('Measurement')
plt.xlabel('Time')
plt.legend(['X_test', 'y_test', 'predicted'], loc='upper left')

def plot_predictions(test, predicted):
    plt.plot(test, color="gray", label="Real Gold Price")
    plt.plot(predicted, color="red", label="Predicted Gold Price")
    plt.title("Gold Price Prediction (USD)")
    plt.xlabel("Time")
    plt.ylabel("Gold Price (USD)")
    plt.legend()
    plt.show()

plot_predictions(y_test[:, 0], predicted_gold_price[:, 0])
```

4. Experiments Report

Provide a summary of experimental results, explain the meaning of your result and how your model can be used to address the related business problem.

```
```